Rapport de RODD

Première Partie - Protection de la biodiversité

Adrien BLASSIAU
Corentin JUVIGNY

19 mars 2020

Table des matières

Introduction

Ce rendu regroupe l'ensemble des rapports associés aux 4 projets réalisés dans le cadre de la première partie de l'UE **Recherche Opérationnelle et Développement Durable** (RODD) encadrée par **Amélie Lambert** :

— un projet sur la sélection de réserves naturelles.

— un projet sur la maîtrise des effets néfastes engendrés par la fragmentation du paysage.

— un projet sur la protection de la diversité génétique.

— un projet sur l'exploitation durable de la forêt.

Nous présenterons les **modèles** ainsi que l'**analyse des résultats obtenus** sur les intances fournies et des instances générées aléatoirement.

1 Projet 1 - Sélection de réserves naturelles

1.1 Présentation du problème

On souhaite assurer à chaque espèce ou site menacés un espace ou son avenir est garanti.
Pour cela, on s'intéresse à un **ensemble d'espèces à protéger** $\mathrm{E} = \{e_1, e_2, \ldots, e_p\}$ vivant sur un **ensemble de parcelles** $\mathrm{S} = \{s_1, s_2, \ldots, s_n\}$ répartis sur un territoire.

L'**objectif** est de déterminer un sous-ensemble de parcelles de coût minimal et tel que, pour chaque espèce e_k, la probabilité de présence dans la réserve, c'est-à-dire dans ce sous-ensemble de parcelles, soit supérieure ou égale à une valeur donnée α_k.

1.2 Modélisation du problème

On obtient le programme non linéaire suivant ;

$$\min \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^{p} c_{ij}$$

$$\text{s.c.} \quad \sum_{i=1}^{m} x_{i1}^{c} = 0 \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{m} x_{in}^{c} = 0 \qquad \textbf{(C2)}$$

$$\sum_{j=1}^{n} x_{1j}^{c} = 0 \qquad \textbf{(C3)}$$

$$\sum_{j=1}^{n} x_{mj}^{c} = 0 \qquad \textbf{(C4)}$$

$$9x_{ij}^{c} \leq \sum_{a=i-1}^{i+1} \sum_{b=i-1}^{i+1} x_{ab}^{p} \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] \qquad \textbf{(C5)}$$

$$1 - \prod_{i=1}^{m} \prod_{j=1}^{n} 1 - p_{kij} x_{ij}^{c} \geq \alpha_k \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p]\!] \qquad \textbf{(C6)}$$

$$1 - \prod_{i=1}^{m} \prod_{j=1}^{n} 1 - p_{kij} x_{ij}^{p} \geq \alpha_k \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![p+1;p+q]\!] \qquad \textbf{(C7)}$$

$$x_{ij}^{c} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$$

$$x_{ij}^{p} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$$

$$\alpha_k \in [0,1] \qquad \forall k \in [\![1;p+q]\!]$$

$$p_{ijk} \in [0,1] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p+q]\!]$$

On obtient le programme linéaire suivant, après linéarisation par le logarithme du programme précédent :

$$\min \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^{p} c_{ij}$$

s.c.
$$\sum_{i=1}^{m} x_{i1}^{c} = 0 \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{m} x_{in}^{c} = 0 \qquad \textbf{(C2)}$$

$$\sum_{j=1}^{n} x_{1j}^{c} = 0 \qquad \textbf{(C3)}$$

$$\sum_{j=1}^{n} x_{mj}^{c} = 0 \qquad \textbf{(C4)}$$

$$9x_{ij}^{c} \leq \sum_{a=i-1}^{i+1} \sum_{b=i-1}^{i+1} x_{ab}^{p} \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] \qquad \textbf{(C5)}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \log(1-p_{kij}) x_{ij}^{c} \leq \log(1-\alpha_k) \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p]\!] \qquad \textbf{(C6)}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \log(1-p_{kij}) x_{ij}^{p} \leq \log(1-\alpha_k) \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![p+1;p+q]\!] \qquad \textbf{(C7)}$$

$$x_{ij}^{c} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$$

$$x_{ij}^{p} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$$

$$\alpha_k \in [0,1] \qquad \forall k \in [\![1;p+q]\!]$$

$$p_{ijk} \in [0,1] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p+q]\!]$$

Ce **programme linéaire en variable 0-1** modélise le problème de la détermination d'une réserve optimale avec zone centrale et zone tampon. On utilise **deux variables de décision** :

— x_{ij}^{p} prend 1 si la case de coordonnées (i,j) est **protégée**, 0 sinon.

— x_{ij}^{c} prend 1 si la case de coordonnées (i,j) est **centrale**, 0 sinon.

Les **contraintes** ont les significations suivantes :

— Les contraintes **C1** à **C4** formalise le fait que les cases bordant la zone d'étude **ne peuvent pas être centrales**.

— La contrainte **C5** formalise le fait que pour être **centrale**, une case **doit être entourée par 8 cases protégées**.

— La contrainte **C6** formalise le fait que la probabilité de présence d'une **espèce en danger** doit être **supérieure à un seuil fixé**.

— La contrainte **C7** formalise le fait que la probabilité de présence d'une **espèce commune** doit être **supérieure à un seuil fixé**.

1.3 Résolution du problème

On résout le problème en considérant 3 espèces rares numérotées de 1 à 3 et trois espèces communes numérotées de 4 à 6. On utilise la matrice des coûts du Tableau 1 et on étudie les 4 cas suivants :

— Cas n°1 : $\alpha_k = 0.5$ $(k = 1,\ldots,6)$

— Cas n°2 : $\alpha_k = 0.9$ $(k = 1,\ldots,3)$ et $\alpha_k = 0.5$ $(k = 4,\ldots,6)$

— Cas n°3 : $\alpha_k = 0.5$ $(k = 1,\ldots,3)$ et $\alpha_k = 0.9$ $(k = 4,\ldots,6)$

— Cas n°4 : $\alpha_k = 0.8$ $(k = 1,\ldots,3)$ et $\alpha_k = 0.6$ $(k = 4,\ldots,6)$

6	6	6	4	4	4	4	8	8	8
6	6	6	4	4	4	4	8	8	8
6	6	6	4	4	4	4	8	8	8
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
4	4	4	6	6	6	6	5	5	5
4	4	4	6	6	6	6	5	5	5
4	4	4	6	6	6	6	5	5	5

TABLE 1 – Matrice des coûts utilisées

On obtient les résultats visibles sur le Tableau 2, le Tableau 3, le Tableau 4 et le Tableau 5 qui **correspondent aux résultats attendus**. Un récapitulatif des données voulues est donné dans le Tableau 6.

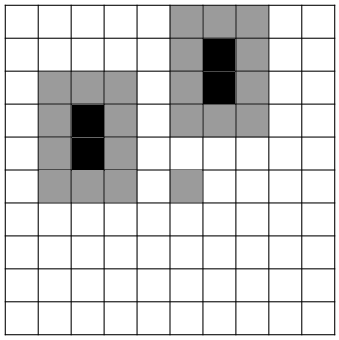

TABLE 2 – Résultats obtenus pour le **Cas n°1** : **coût = 119** et **probabilités de survies : (0.58, 0.52, 0.64, 0.86176, 0.52, 0.755)**.

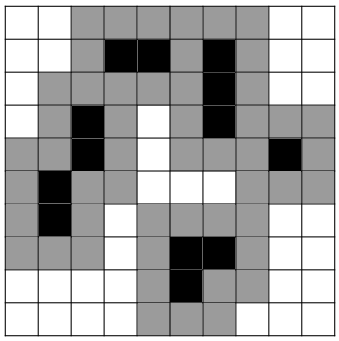

TABLE 3 – Résultats obtenus pour le **Cas n°2** : **coût = 327** et **probabilités de survies : (0.915328, 0.90784, 0.91936, 0.980491571, 0.892, 0.9814785)**.

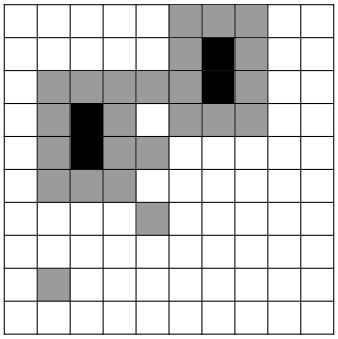

TABLE 4 – Résultats obtenus pour le **Cas n°3** : **coût = 130** et **probabilités de survies : (0.58, 0.52, 0.64, 0.9336448, 0.91, 0.9265)**.

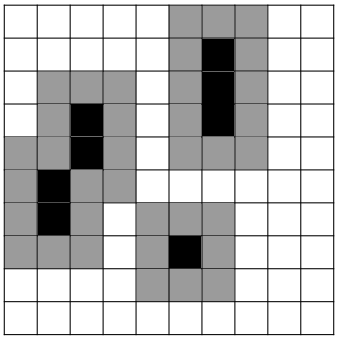

TABLE 5 – Résultats obtenus pour le **Cas n°4** : **coût = 211** et **probabilités de survies : (0.8236, 0.808, 0.82, 0.972130816, 0.784, 0.8775)**.

	temps (s)	nombre de noeuds	coût	probabilité de survie
Cas n°1	0,09	0	119	(0.58, 0.52, 0.64, 0.86176, 0.52, 0.755)
Cas n°2	0,01	0	327	(0.915328, 0.90784,0.91936, 0.980491571, 0.892, 0.9814785)
Cas n°3	0,12	0	130	(0.58, 0.52, 0.64,0.9336448, 0.91, 0.9265)
Cas n°4	0,06	0	211	(0.8236, 0.808, 0.82, 0.972130816, 0.784, 0.8775)

TABLE 6 – Résultats obtenus pour les différents cas demandés.

1.4 Étude de l'impact de la taille d'une instance sur le temps moyen de calcul

On étude l'**impact de la taille d'une instance sur le temps moyen de calcul**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— Les **coûts** sont choisis **entre 0 et 10 compris**.

— Les **probabilités** sont choisies **uniformément entre 0 et 1**.

— Les α sont choisies **uniformément entre 0 et 1**.

— On choisit **3** espèces **communes** et **3** espèces en **danger**.

— On fait varier les tailles de 10 en 10 en gardant $m = n$ en s'arrêtant à 100.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse donc au temps **moyen** (Figure 12) ainsi qu'à la répartition des valeurs obtenues (Figure 14) en fonction de la taille de l'instance.

On remarque que **plus la taille des instances est importante, plus le temps de calcul en moyenne est élevé et plus les écarts de temps de calcul sont importants**. L'évolution semble **exponentielle**. En passant au log (Figure 13), on observe que l'évolution est presque linéaire. Il aurait fallut faire plus d'expériences pour avoir un résultat plus précis. On peut tout de même imaginer que le problème est **difficile**.

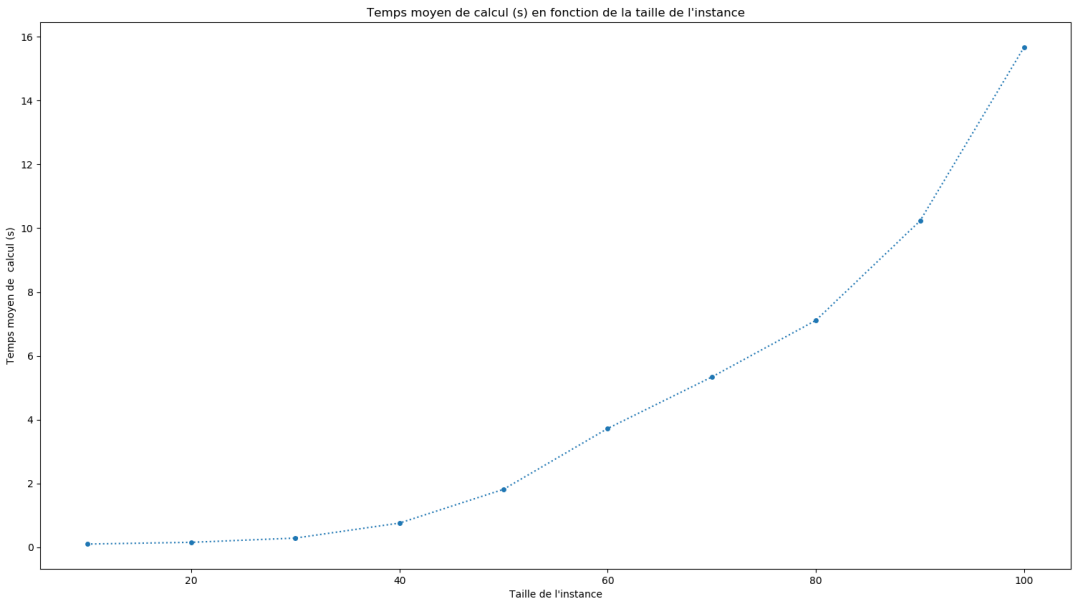

FIGURE 1 – Temps moyen de calcul (s) en fonction de la taille de l'instance

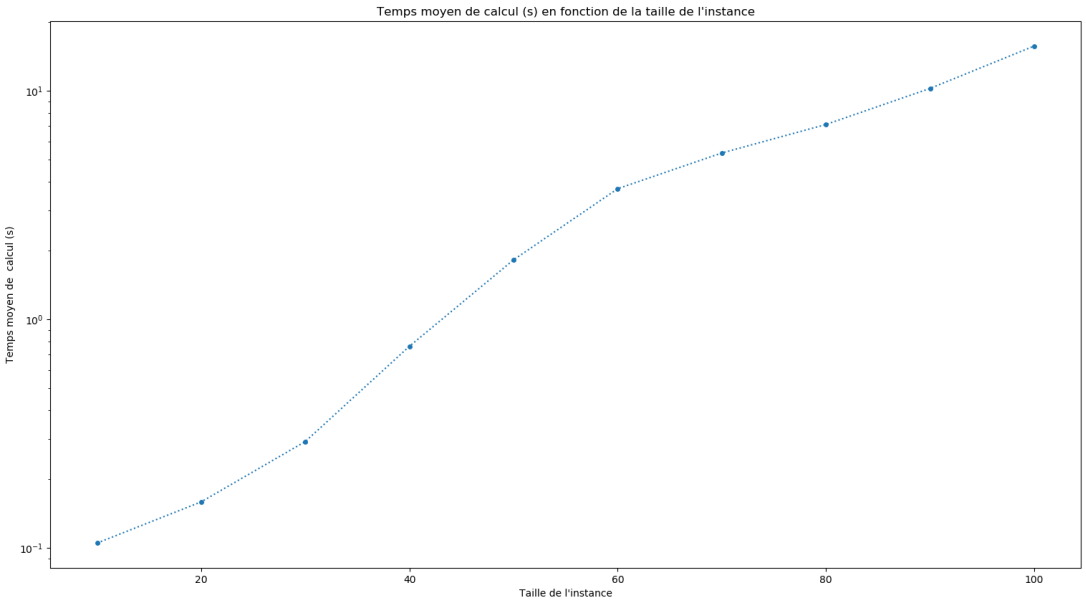

FIGURE 2 – Temps moyen de calcul (s) en fonction de la taille de l'instance

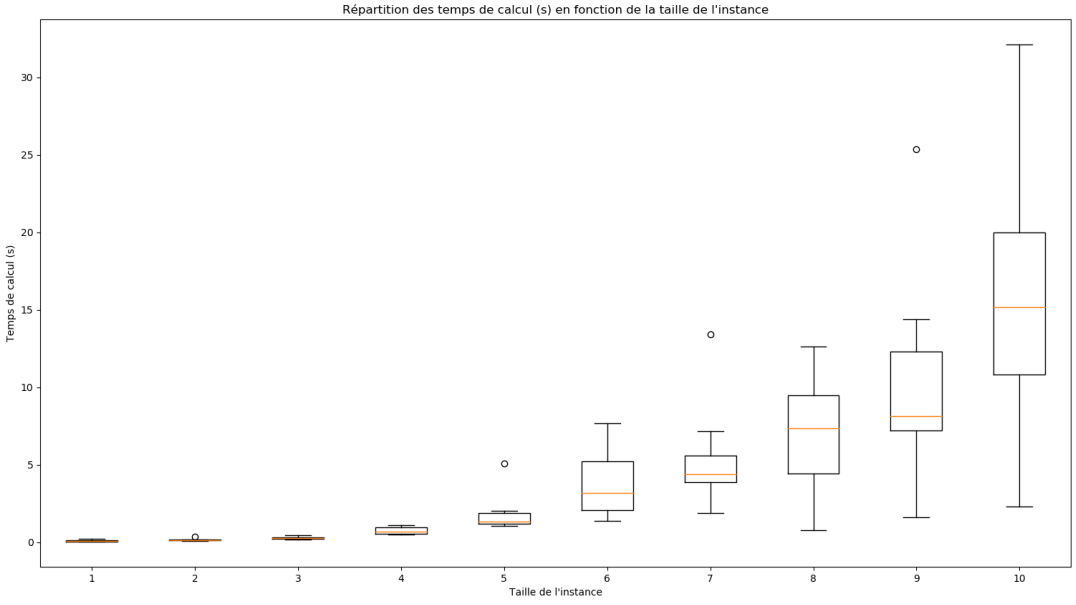

FIGURE 3 – Répartition des temps de calcul (s) en fonction de la taille de l'instance

1.5 Nouveau modèle

2 Projet 2 - Maîtrise des effets néfastes engendrés par la fragmentation su paysage

2.1 Présentation du problème

On souhaite minimiser la fragmentation du paysage, c'est-à-dire l'éclatement de grandes superficies en petites parcelles. Pour cela, on va étudier un indicateur, *la distance moyenne au plus proche voisin (DMPPV)*. On considère un ensemble de parcelles $\mathrm{S} = \{s_1, s_2, \ldots, s_n\}$ répartis sur un territoire d'aire A.

L'**objectif** est de sélectionner un sous-ensemble de parcelles d'aire totale comprise entre A_{min} et A_{max}, de coût total inférieur ou égal à une certaine valeur B et qui minimise *DMPPV*.

2.2 Modélisation du problème

On modélise le problème avec le programme d'optimisation combinatoire factionnaire suivant :

$$\min \quad z = \frac{\sum_{i_1=1}^{m}\sum_{j_1=1}^{n}\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} d_{i_1 j_1 i_2 j_2} y_{i_1 j_1 i_2 j_2}}{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}}$$

$$\text{s.c.} \quad \mathrm{A}_{min} \leq \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \leq \mathrm{A}_{max} \qquad \textbf{(C2)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} 10 x_{ij} c_{ij} \leq \mathrm{B} \qquad \textbf{(C3)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ijij} = 0 \qquad \textbf{(C4)}$$

$$y_{i_1 j_1 i_2 j_2} \leq x_{i_2 j_2} \qquad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!] \qquad \textbf{(C5)}$$

$$\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} y_{i_1 j_1 i_2 j_2} \leq x_{i_1 j_1} \qquad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!] \qquad \textbf{(C6)}$$

$$x_{ij} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$$

$$y_{i_1 j_1 i_2 j_2} \in [\![0;1]\!] \qquad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!]$$

$$\mathrm{A}_{min}, \mathrm{A}_{max}, \mathrm{B} \in \mathrm{N}$$

L'étude de ce **programme d'optimisation** peut se ramener à l'étude du programme linéaire suivant, où l'on pose $f(x,y) = \sum_{i_1=1}^{m}\sum_{j_1=1}^{n}\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} d_{i_1 j_1 i_2 j_2} y_{i_1 j_1 i_2 j_2}$ et $g(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$:

$$\min \quad z = f(x,y) - \lambda g(x,y)$$

s.c. $\mathrm{A}_{min} \leq \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$ **(C1)**

$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \leq \mathrm{A}_{max}$ **(C2)**

$\sum_{i=1}^{m}\sum_{j=1}^{n} 10 x_{ij} c_{ij} \leq \mathrm{B}$ **(C3)**

$\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ijij} = 0$ **(C4)**

$y_{i_1 j_1 i_2 j_2} \leq x_{i_2 j_2} \quad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!]$ **(C5)**

$\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} y_{i_1 j_1 i_2 j_2} \leq x_{i_1 j_1} \quad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!]$ **(C6)**

$x_{ij} \in [\![0;1]\!] \quad \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!]$

$y_{i_1 j_1 i_2 j_2} \in [\![0;1]\!] \quad \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!]$

$\mathrm{A}_{min}, \mathrm{A}_{max}, \mathrm{B} \in \mathrm{N}$

On utilise **deux variables de décision** :

— x_{ij} prend 1 si la case de coordonnées (i,j) est sélectionnée, 0 sinon.

— $y_{i_1 j_1 i_2 j_2}$ prend 1 si les cases de coordonnées (i1,j1) et (i2,j2) sont sélectionnée et si (i2,j2) et la case la plus proche de (i1,j1), 0 sinon.

Les **contraintes** ont les significations suivantes :

— Les contraintes **C1** à **C2** formalise le fait que l'**aire totale** des cases sélectionnée doit être comprise entre A_{min} et A_{max}.

— La contrainte **C3** formalise le fait le **coût total des cases sélectionnées** doit être inférieur à un coût B.

— La contrainte **C4** formalise le fait qu'une **case ne peut pas avoir comme case la plus proche elle même**.

— La contrainte **C5** formalise le fait que si une case de coordonnées (i1,j1) est la plus proche d'une autre case (i2,j2), alors (i2,j2) est sélectionnée.

— La contrainte **C6** formalise le fait que si une case (i1,j1) est la plus proche d'une autre case (i2,j2), alors cette case est unique et la case (i1,j1) doit être sélectionnée.

2.3 Résolution du problème

On résout le problème linéaire plus haut en appliquant l'**algorithme de Dinkelbach**. On utilise la matrice des coûts du Tableau 7 et on étudie les 3 cas suivants :

— Cas n°1 : $\mathrm{A}_{min} = 30$, $\mathrm{A}_{min} = 35,\ \mathrm{B} = 920$

— Cas n°2 : $\mathrm{A}_{min} = 20$, $\mathrm{A}_{min} = 21,\ \mathrm{B} = 520$

— Cas n°3 : $\mathrm{A}_{min} = 70$, $\mathrm{A}_{min} = 75,\ \mathrm{B} = 3500$

7	3	10	10	2	8	6	4	5	5
7	7	10	5	2	8	6	3	9	9
7	3	4	6	3	2	4	9	7	8
6	2	7	6	4	7	5	10	7	8
2	4	3	4	9	6	4	9	8	4
7	5	2	9	8	9	5	6	10	10
5	2	3	7	9	9	4	9	6	3
5	2	9	4	2	8	6	9	3	4
9	6	5	4	5	6	8	9	6	6
8	8	7	7	3	5	8	3	9	9

TABLE 7 – Matrice des coûts utilisées

On obtient les résultats visibles sur le Tableau 8, le Tableau 9 et le Tableau 10 qui **correspondent aux résultats attendus**. Un récapitulatif des données voulues est donné dans le Tableau 11.

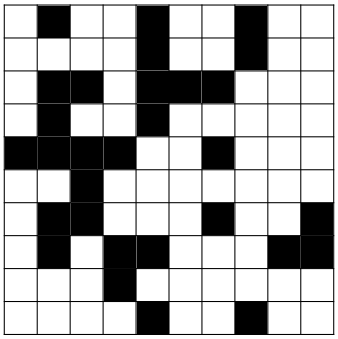

TABLE 8 – Résultats obtenus pour le **Cas n°1** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 30** et **DMPPV = 1.155009385**.

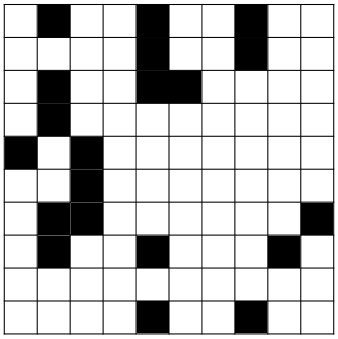

TABLE 9 – Résultats obtenus pour le **Cas n°2** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 20** et **DMPPV = 1.2739354335**.

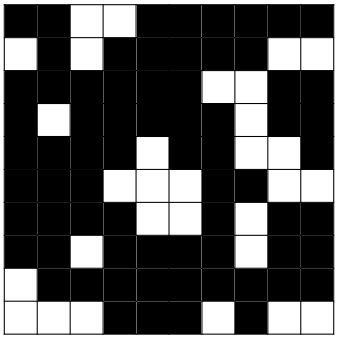

TABLE 10 – Résultats obtenus pour le **Cas n°3** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 71** et **DMPPV = 1**.

	temps (s)	nombre de noeuds	nombre d'itérations	DMPPV	nombre de sélections
Cas n°1	0,41	0	2	1.155009385	30
Cas n°2	0,35	0	2	1.273935433	20
Cas n°3	0,36	0	2	1	71

TABLE 11 – Résultats obtenus pour les différents cas demandés.

2.4 Étude de l'impact de la taille d'une instance sur le temps moyen de calcul

On étude l'**impact de la taille d'une instance sur le temps moyen de calcul**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— A_{min} est choisi uniformément entre $0.2 * m * n$ et $0.8 * m * n$.

— A_{max} est choisi uniformément entre $A_{min} + 1$ et $A_{max} + 10$.

— B est choisi uniformément entre $10 * m * n$ et $50 * m * n$.

— On fait varier les tailles de 5 en 5 en gardant $m = n$ et en s'arrêtant à 30.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse donc au temps **moyen** (Figure 4) ainsi qu'à la répartition des valeurs obtenues (Figure 6) en fonction de la taille de l'instance.

On remarque que **plus la taille des instances est importante, plus le temps de calcul en moyenne est élevé et plus les écarts de temps de calcul sont importants**. L'évolution semble **exponentielle**. En passant au log (Figure 5), on observe que l'évolution est presque linéaire. Il aurait fallut faire plus d'expériences pour avoir un résultat plus précis. On peut tout de même imaginer que le problème est **difficile**.

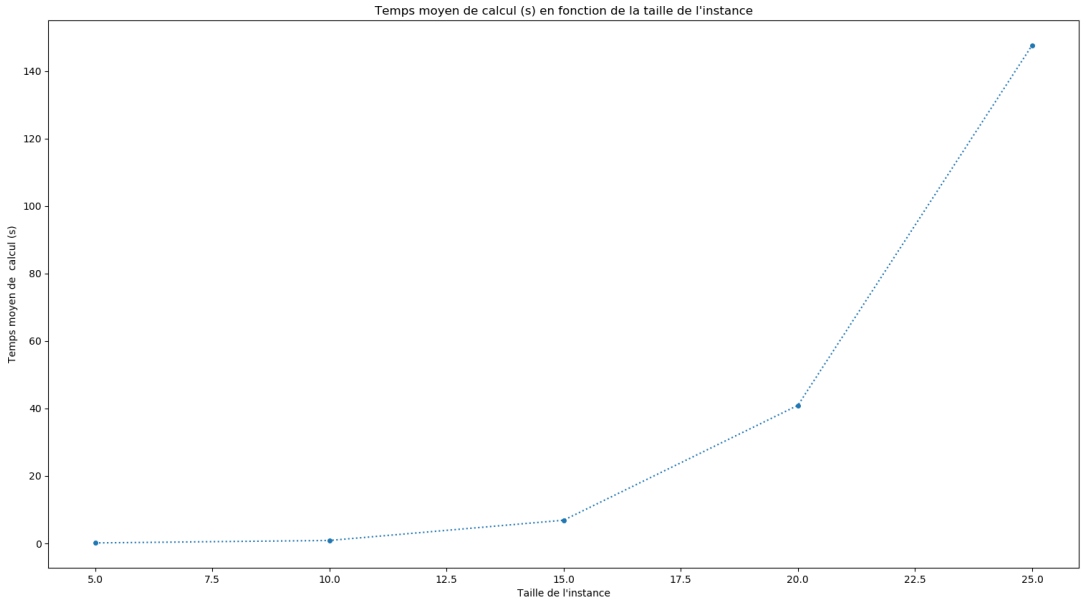

FIGURE 4 – Temps moyen de calcul (s) en fonction de la taille de l'instance

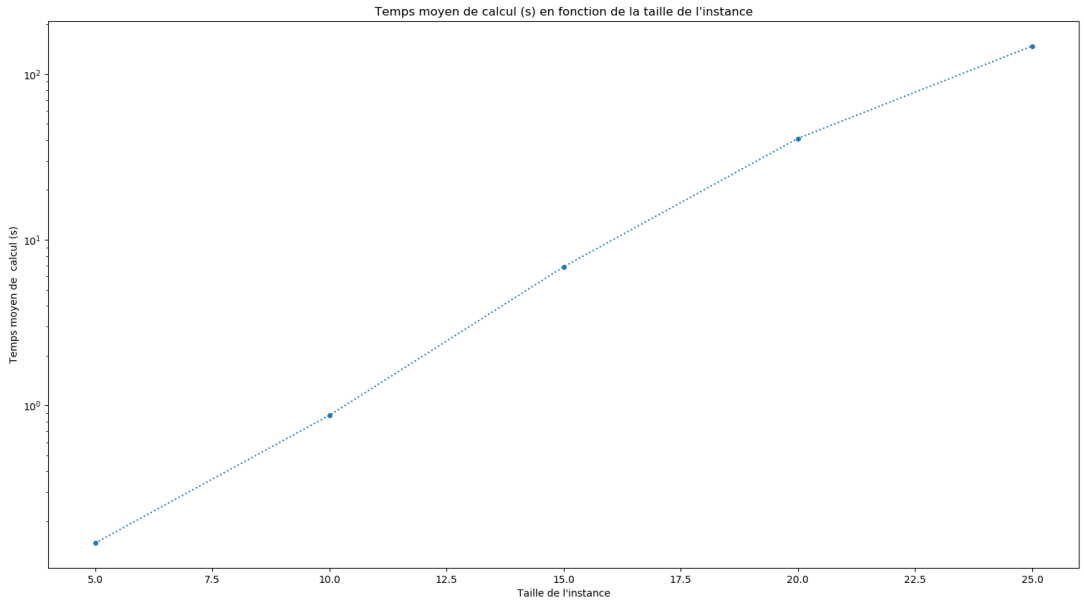

FIGURE 5 – Temps moyen de calcul (s) en fonction de la taille de l'instance

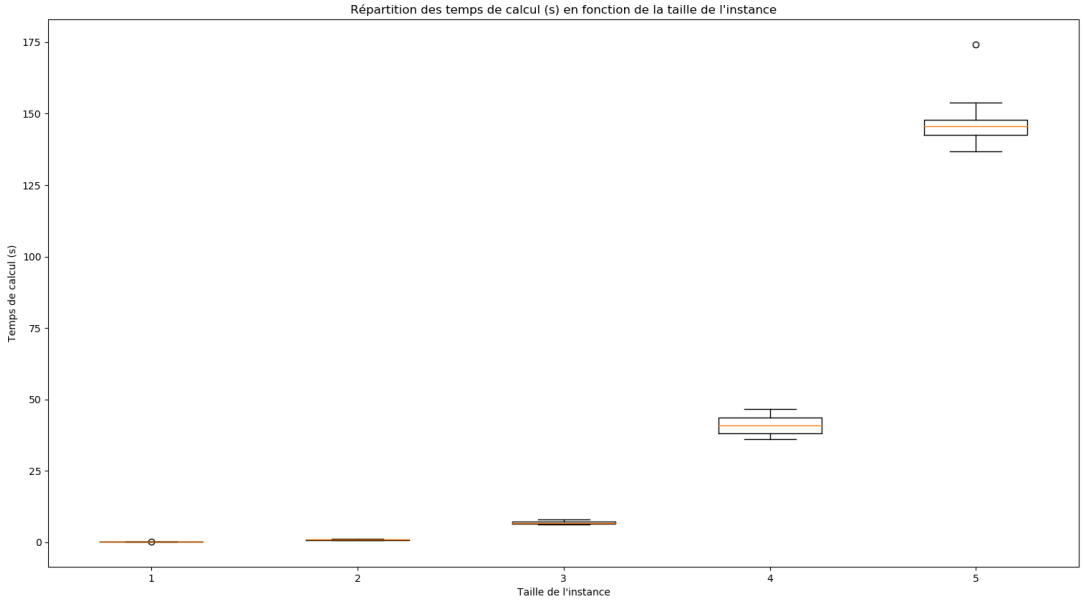

FIGURE 6 – Répartition des temps de calcul (s) en fonction de la taille de l'instance

3 Projet 3 - Protection de la diversité génétique

3.1 Présentation du problème

Le problème étudié est un problème classique de la conservation génétique. Il consiste à déterminer la contribution optimale de chaque individu d'un population de façon à minimiser la perte d'allèles dans la population engendrée. Pour cela, on considère une population formée de P individus $\{\mathrm{I}_1, \mathrm{I}_2, \ldots, \mathrm{I}_p\}$ appartenant à une espèce donnée et un ensemble de M gènes polyallèliques à prendre en compte $\{g_1, g_2, \ldots, g_\mathrm{M}\}$. Pour chaque gène g_i, les allèles possibles sont $\{a_{i_1}, a_{i_2}, \ldots, a_{i_{t(i)}}\}$.

L'**objectif** est de minimiser l'espérance mathématique du nombre d'allèles disparus dans la génération suivante tout en conservant une population de taille constante.

3.2 Modélisation du problème

On modélise le problème avec le programme d'optimisation combinatoire non linéaire suivant suivant :

$$\min \quad z = \sum_{j=1}^{\mathrm{G}} \sum_{k=1}^{\mathrm{A}} z_{jk}$$

$$\text{s.c.} \quad \sum_{i=1}^{\mathrm{N}_m} x_i = \sum_{i=\mathrm{N}_m+1}^{\mathrm{N}_m+\mathrm{N}_f} x_i \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{\mathrm{N}} x_i = 2 * \mathrm{N} \qquad \textbf{(C2)}$$

$$z_{jk} \geq t_{jk} - \sum_{i=1, p_{ijk}=0}^{\mathrm{N}} x_i \qquad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C3)}$$

$$t_{jk} \geq \prod_{i=1, p_{ijk}\neq 0}^{\mathrm{N}} 0.5^{x_i} \qquad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C4)}$$

$$0 \leq x_i \leq 2 \qquad \forall i \in [\![1;\mathrm{N}]\!] \qquad \textbf{(C5)}$$

$$y_{jk} \geq 0, z_{jk} \geq 0 \qquad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C6)}$$

$$\mathrm{G}, \mathrm{A}, \mathrm{N}_m, \mathrm{N}_f, \mathrm{N} \in \mathbb{N}$$

L'étude de ce **programme d'optimisation** peut se ramener à l'étude du programme linéaire suivant, en passant le produit à la fonction logarithmique puis en l'approchant par une fonction linéaire par morceaux, comme montré dans l'énoncé :

$$\min \quad z = \sum_{j=1}^{\mathrm{G}} \sum_{k=1}^{\mathrm{A}} z_{jk}$$

s.c.
$$\sum_{i=1}^{\mathrm{N}_m} x_i = \sum_{i=\mathrm{N}_m+1}^{\mathrm{N}_m+\mathrm{N}_f} x_i \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{\mathrm{N}} x_i = 2 * \mathrm{N} \qquad \textbf{(C2)}$$

$$z_{jk} \leq t_{jk} - \sum_{i=1, p_{ijk}=0}^{\mathrm{N}} x_i \qquad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C3)}$$

$$ln(\theta_r) + \frac{t_{jk} - \theta_r}{\theta_r} \geq \sum_{i=1, p_{ijk}\neq 0}^{\mathrm{N}} x_i * ln(0.5) \qquad \forall r \in [\![1;\mathrm{T}]\!], \quad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C4)}$$

$$0 \leq x_i \leq 2 \qquad \forall i \in [\![1;\mathrm{N}]\!] \qquad \textbf{(C5)}$$

$$y_{jk} \geq 0, z_{jk} \geq 0 \qquad \forall j \in [\![1;\mathrm{G}]\!], \quad \forall k \in [\![1;\mathrm{A}]\!] \qquad \textbf{(C6)}$$

$$\mathrm{G}, \mathrm{A}, \mathrm{N}_m, \mathrm{N}_f, \mathrm{N} \in \mathbb{N}$$

On pose $\theta_r = \theta_1^{\frac{\mathrm{T}-r}{\mathrm{T}-1}}$ avec $r = 0.001$ et $\mathrm{T} = 50$. On remarque que p_{ijk} correspond à la probabilité de disparition de l'allèle k du gène j de l'individu i. Les autres constantes sont données dans l'énoncé et dans les fichiers de données.

On utilise **trois variables de décision** :

— x_i correspond au nombre d'enfant de l'individu i à la génération suivante.

— y_{jk} correspond à la probabilité de disparition de l'allèle k positionné sur le gène j.

— z_{jk} correspond est une variable intermédiaire permettant de représenter tout les cas possibles

Les **contraintes** ont les significations suivantes :

— La contrainte **C1** formalise le fait que les mâles et les femelles doivent avoir le même nombre d'enfants à la génération suivante (on part du principe qu'il faut un mâle et une femelle pour se reproduire au sein de cette espèce).

— La contrainte **C2** formalise le fait que l'on conserve une population de taille constante d'une génération à l'autre.

— Les contraintes **C3** et **C4** sont liées au calcul des probabilité de disparition.

3.3 Résolution du problème

On résout le problème linéaire plus haut sur l'instance fournie. Elle est composée de $\mathrm{N} = 8$ individus, $\mathrm{N}_m = 4$ mâles, $\mathrm{N}_f = 4$ femelles, $\mathrm{C} = 1$ paire de chromosome, $\mathrm{G} = 5$ gènes par chromosomes et $\mathrm{A} = 2$ allèles par gènes. On s'intéresse à deux cas différents :

— Cas n°1 : $x_i \leq 3$, $\forall i \in [\![1;8]\!]$

— Cas n°2 : $x_i = 2$, $\forall i \in [\![1;8]\!]$

On obtient les solutions suivantes, présente sur le Tableau 12.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Cas n°1	1	3	3	1	3	3	2	0
Cas n°2	2	2	2	2	2	2	2	2

TABLE 12 – Solutions obtenues pour le **Cas n°1** et le **Cas n°2**

Les résultats détaillés demandés dans l'énoncé sont présents dans le Tableau 13.

	temps (s)	nombre de noeuds	proba de disparition	E(nombre d'allèle dsparus)	borne inf
Cas n°1	0.03	0	b=0.015625,reste=0	0.015625	0,0155881
Cas n°2	0.00	0	b=0.0625,reste=0	0.015625	0.0624334

TABLE 13 – Résultats obtenus pour le **Cas n°1** et le **Cas n°2**

3.4 Étude de l'impact de certains paramètres

Étude de l'impact du nombre de morceaux de l'approximation sur la borne inférieure et le temps de calcul

On étude tout d'abord l'**impact du nombre de morceaux considérés dans la fonction linéaire par morceaux** approximant la fonction logarithmique **sur la borne inférieure obtenue** et **du temps de calcul**. Pour cette étudie, on n'utilise aucun générateur d'instance aléatoire. L'instance étudiée est celle du sujet, on se place dans le **Cas n°1**.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. On s'intéresse à l'évolution de la borne inférieure en fonction du nombre de morceaux sur la Figure 7, l'évolution du gap associé Figure 8 et l'évolution du temps moyen de calcul (s) en fonction du nombre de morceaux sur la Figure 9. Un résumé des données obtenues est présent sur le Tableau 14.

On remarque que **plus le nombre de morceaux** permettant d'approximer la fonction logarithmique **est important**, **plus on se rapproche de l'objectif**, c'est-à-dire de l'espérance mathématique du nombre d'allèles disparus dans la génération suivante, de **0.015625**. **Le gap se resserre**, l'approximation est de plus en plus fine.

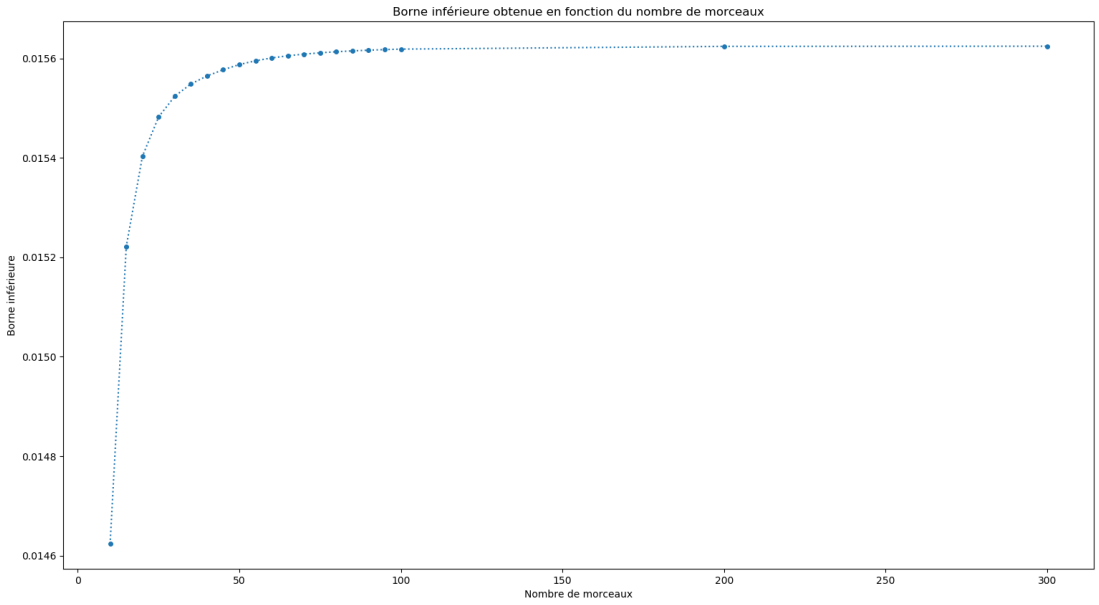

FIGURE 7 – Borne inférieure obtenue en fonction du nombre de morceaux

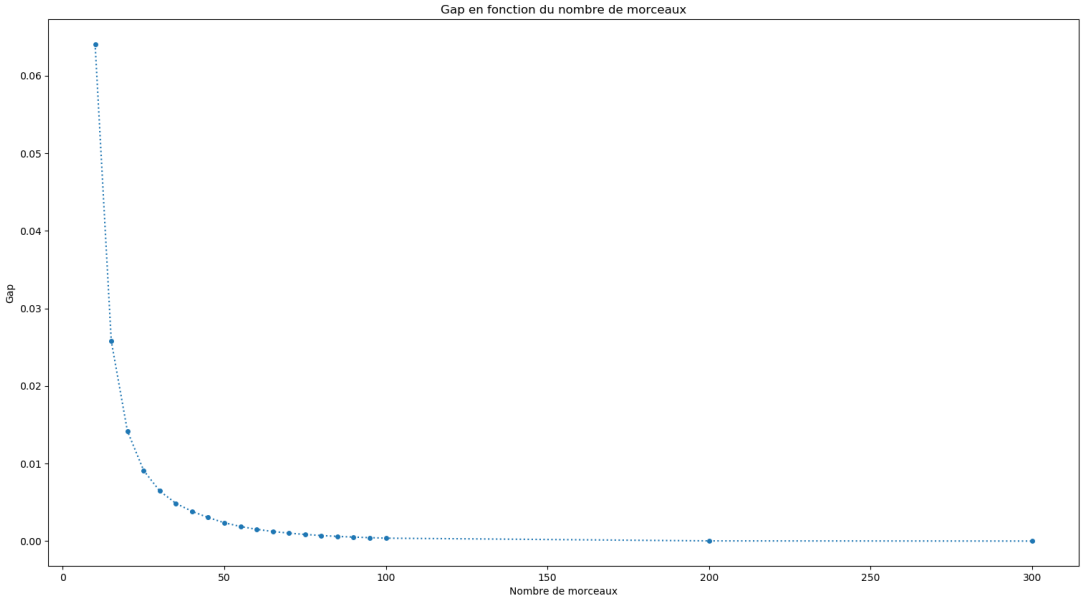

FIGURE 8 – Gap obtenu en fonction du nombre de morceaux

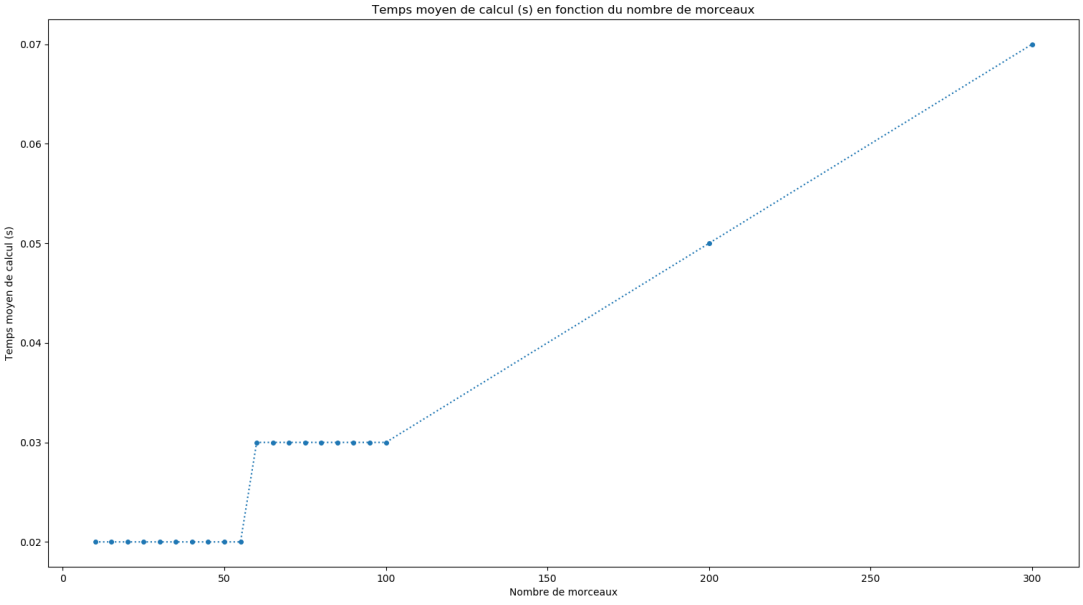

FIGURE 9 – Temps moyen de calcul (s) en fonction du nombre de morceaux

nombre de morceaux	borne inféreure	temps de calcul (s)	gap
10	0.0146234137194948	0.02	0.06410152195233276
15	0.0152217299304239	0.02	0.02580928445287045
20	0.0154033371394141	0.02	0.014186423077497623
25	0.015482426604255	0.02	0.009124697327679954
30	0.0155241810472989	0.02	0.006452412972870358
35	0.0155490488914568	0.02	0.004860870946764795
40	0.0155651379766199	0.02	0.0038311694963264475
45	0.0155776267085092	0.02	0.003031890655411207
50	0.0155880964533051	0.02	0.0023618269884736476
55	0.0155957036448688	0.02	0.0018749667283968208
60	0.01560138048123	0.03	0.0015116492012799965
65	0.0156057116773627	0.03	0.0012344526487871876
70	0.0156090784129599	0.03	0.0010189815705663463
75	0.0156117373864284	0.03	0.0008488072685823855
80	0.0156138663293322	0.03	0.0007125549227392503
85	0.0156155912837706	0.03	0.0006021578386815607
90	0.0156170035246442	0.03	0.0005117744227711718
95	0.0156181703782267	0.03	0.0004370957934911601
100	0.0156191423236521	0.03	0.0003748912862655551
200	0.0156246614385874	0.05	2.1667930406366054e-05
300	0.0156249989405914	0.07	6.780215044965843e-08

TABLE 14 – Résultats obtenus

Étude de l'impact de la taille de la population sur la solution obtenue

On étude maintenant d'abord l'**impact de la taille de la population sur le nombre moyen d'allèles qui disparait**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— La taille N de la population est donnée en entrée du programme

— L'instance est composée de $\mathrm{N}_m = \frac{\mathrm{N}}{2}$ mâles, $\mathrm{N}_f = \frac{\mathrm{N}}{2}$ femelles, $\mathrm{C} = 1$ paire de chromosome, $\mathrm{G} = 5$ gènes par chromosomes et $\mathrm{A} = 2$ allèles par gènes. On choisit aléatoirement entre 0 ou 1 les allèles présents sur chaque gène.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse l'évolution de l'espérance du nombre d'allèles disparus en fonction de la taille de l'instance sur la Figure 10. La répartition des valeurs est donnée en Figure 11.

On remarque que **plus la taille de la population est importante**, moins le risque de perte génétique augmente, donc **plus la diversité génétique est conservée**.

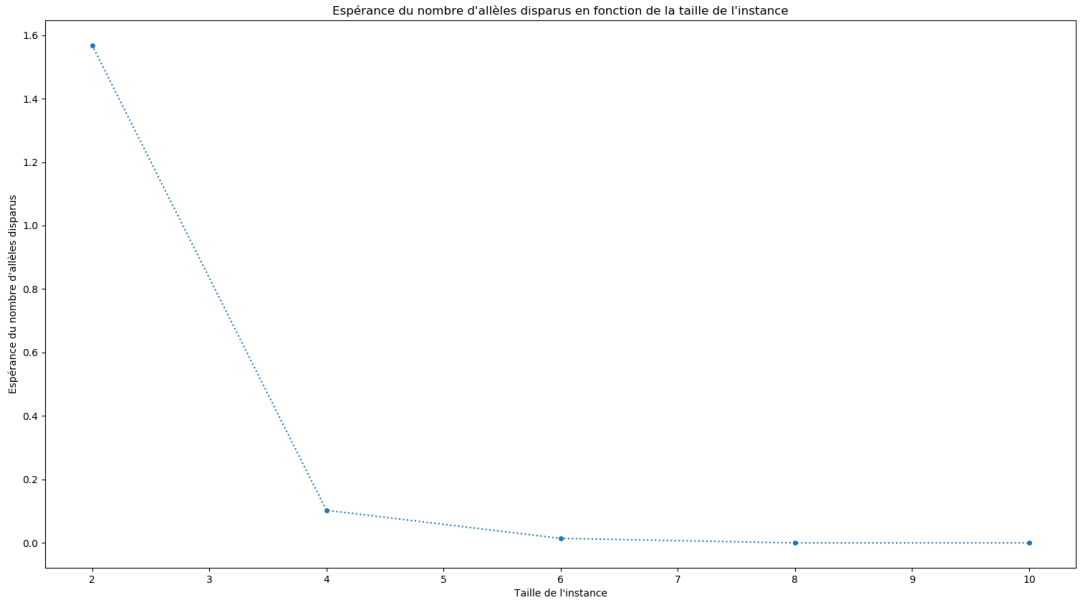

FIGURE 10 – Espérance du nombre d'allèles disparus en fonction de la taille de l'instance

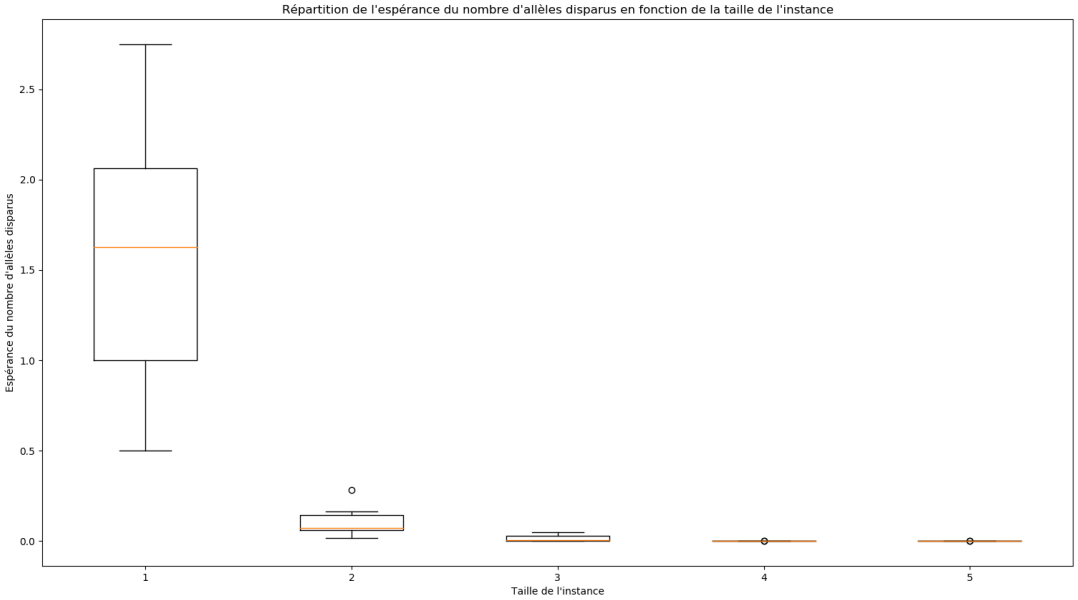

FIGURE 11 – Répartition de l'espérance du nombre d'allèles disparus en fonction de la taille de l'instance

4 Projet 4 - Exploitation durable de la forêt

4.1 Présentation du problème

On souhaite exploiter les forêts **de manière durable**, c'est-à-dire en protégeant le mieux possible certaines espèces.
Pour cela, on considère un ensemble de **parcelles forestières carrées** et identiques représenté par une matrice m*n et **deux types d'espèces** e_1 et e_2.
Deux types de parcelle existent : les parcelles coupées et les parcelles non coupées.
L'espèce e_1 vit dans les **parcelles coupées** tandis que l'espèce e_2 vit **en lisière entre les parcelles coupées et les parcelles non coupées**.
La **population attendue** de l'espèce e_1 dans chaque parcelle s_{ij} coupée (resp. non coupée) est t_{ij} (resp. 0).
La **population attendue** de l'espèce e_2 est $g\mathrm{L}$ où g désigne la population attendue de l'espèce e_2 pour chaque kilomètre de lisière et L la longueur totale de lisière compte tenu des coupes réalisées.

L'**objectif** est de déterminer les parcelles à couper et les parcelles à laisser en l'état de façon à **maximiser la somme pondérée des populations des deux espèces**.

4.2 Première modélisation du problème (P1)

Une première modélisation **(P1)** du problème est proposée dans le sujet. C'est une formulation par un programme linéaire en variables mixtes.

$$
\begin{aligned}
\max \quad & z = w_1 \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} t_{ij}(1-x_{ij}) + w_2 g l \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} 4x_{ij} - d_{ij} \\
\text{s.c.} \quad & d_{ij} \geq \sum_{(k,l)\in \mathrm{A}_{ij}} x_{kl} - |\mathrm{A}_{ij}|(1-x_{ij}) && \forall (i,j) \in \mathrm{M}\times\mathrm{N} \quad \textbf{(C1)} \\
& d_{ij} \in \mathbb{R}_+^{\mathrm{M}\times\mathrm{N}} && \forall (i,j) \in \mathrm{M}\times\mathrm{N} \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in \mathrm{M}\times\mathrm{N}
\end{aligned}
$$

où M = $1,\ldots,m$, N = $1,\ldots,n$, w_1 et w_2 sont les coefficients de pondération, l est la longueur du côté de chaque parcelle et A_{ij} désigne l'ensemble des couples (k,l) tels que la parcelle s_{kl} est adjacente à la parcelle s_{ij}.

On utilise **deux variables** :

- x_{ij} est une **variable de décision** qui prend 1 si et seulement si la parcelle s_{ij} est non coupée, 0 sinon.
- d_{ij} est une **variable intermédiaire** qui prend le nombre de parcelles non coupées se trouvant autour d'une parcelle non coupée. Plus précisément, la contrainte **(C1)** peut être comprise ainsi :
  - quand s_{ij} est non coupée, x_{ij} vaut 1, donc le terme $|\mathrm{A}_{ij}|(1-x_{ij})$ vaut 0 et $d_{ij} \geq \sum_{(k,l)\in \mathrm{A}_{ij}} x_{kl}$ correspond au nombre de parcelles non coupées dans les voisins de la parcelle non coupée s_{ij}
  - quand s_{ij} est coupée, x_{ij} vaut 0, donc le terme $\sum_{(k,l)\in \mathrm{A}_{ij}} x_{kl} - |\mathrm{A}_{ij}|(1-x_{ij})$ est négatif, donc d_{ij} est mis à 0.

L'**objectif** peut être compris ainsi :

- le terme $w_1 \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} t_{ij}(1-x_{ij})$ est associé au comptage pondéré par w_1 de la population de l'espèce e_1 sur la parcelle s_{ij} :
  - si $x_{ij} = 0$, la parcelle est coupée et on retrouve que la population attendue de l'espèce e_1 vaut s_{ij}.

— sinon, $x_{ij} = 1$, la parcelle est non coupée et on retrouve que la population attendue vaut 0.

— le terme $w_2 gl \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} 4x_{ij} - d_{ij}$ est associé au comptage pondéré par w_2 de l'espèce e_2 sur la parcelle s_{ij} :

— si $x_{ij} = 0$, la parcelle est coupée, on sait que dans ce cas d_{ij} vaut 0 donc le terme est nul.

— sinon, $x_{ij} = 1$, la parcelle est non coupée et d_{ij} correspond au nombre de parcelles non coupées dans les voisins de la parcelle coupée s_{ij}. Ainsi le terme $\sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} 4x_{ij} - d_{ij}$ correspond au nombre de parcelles coupées entourant s_{ij}. En multipliant ce terme par gl, on obtient bien la population attendue de l'espèce e_2 en lisière de s_{ij}.

4.3 Deuxième modélisation du problème (P2)

On modélise maintenant le problème par un programme quadratique en variable 0-1 noté **(P2)** :

$$\begin{aligned} \max \quad & z = w_1 \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} t_{ij}(1 - x_{ij}) + w_2 gl \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} \sum_{(k,l)\in \mathrm{A}_{ij}} x_{ij}(1 - x_{kl}) \\ \text{s.c.} \quad & x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N} \end{aligned}$$

4.4 Linéarisation de la deuxième modélisation en (P3)

On obtient le programme linéaire en variables mixtes **(P3)** suivant, après linéarisation classique de Fortet de **(P2)** en remplaçant $x_{ij}x_{kl}$ par y_{ijkl} et en rajoutant les contraintes classiques.

$$\begin{aligned} \max \quad & z = w_1 \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} t_{ij}(1 - x_{ij}) + w_2 gl \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} \sum_{(k,l)\in \mathrm{A}_{ij}} (x_{ij} - y_{ijkl}) \\ \text{s.c.} \quad & y_{ijkl} \geq x_i + x_j - 1 \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N}, \quad \forall (k,l) \in \mathrm{A}_{ij} \quad \textbf{(C1)} \\ & y_{ijkl} \leq x_{ij} \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N}, \quad \forall (k,l) \in \mathrm{A}_{ij} \quad \textbf{(C2)} \\ & y_{ijkl} \leq x_{kl} \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N}, \quad \forall (k,l) \in \mathrm{A}_{ij} \quad \textbf{(C3)} \\ & y_{ijkl} \geq 0 \qquad \forall (i,j,k,l) \in \mathrm{M}\times\mathrm{N} \\ & x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N} \end{aligned}$$

On remarque que les contraintes **(C2)** et **(C3)** ne sont pas nécessaires car on **maximise** et que les **coefficients** devant les y_{ijkl} dans la fonction objectif sont **négatifs** (-1). Donc on prend la plus petite valeur possible pour chaque y_{ijkl} d'où l'inutilité de ces deux contraintes.

On obtient la matrice des contraintes suivantes :

$$\mathrm{M} = \begin{array}{c} \begin{matrix} x_{11} & x_{12} & & x_{ij} & & x_{kl} & & y_{1112} & & y_{ijkl} & \end{matrix} \\ \begin{pmatrix} 1 & 1 & \cdots & 0 & \cdots & 0 & \cdots & -1 & \cdots & 0 & \cdots \\ 0 & 0 & \cdots & 1 & \cdots & 1 & \cdots & 0 & \cdots & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

La matrice M est composée de **deux blocs** :

— le premier bloc est la matrice d'incidence sommets-arêtes d'un graphe biparti donc il est **TU**

— le deuxième bloc correspond à moins la matrice identité donc il est **TU**

D'après le cours d'**Optimisation dans les Graphes** (avec Cédric Bentz) si une matrice est TU, alors la juxtaposition de cette matrice avec la matrice identité est aussi une matrice TU. Ainsi, la matrice des contraintes M de **(P3)** est bien **TU**.
Enfin les seconds membres de **(P3)** sont tous positifs.

Bilan : On sait que si un programme linéaire à une matrice des contraintes TU et des seconds membres entiers, alors toute solution de base de ce PL est entière (c'est donc vrai, en particulier, pour toute solution de base optimale).

Ainsi, la solution optimale de **(P3)** est entière et est égale à celle de sa relaxation linéaire **(P4)** :

$$
\begin{aligned}
\max \quad & z = w_1 \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} t_{ij}(1-x_{ij}) + w_2 g l \sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} \sum_{(k,l)\in \mathrm{A}_{ij}} (x_{ij} - y_{ijkl}) \\
\text{s.c.} \quad & y_{ijkl} \geq x_i + x_j - 1 \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N}, \quad \forall (k,l) \in \mathrm{A}_{ij} \qquad \textbf{(C1)} \\
& y_{ijkl} \geq 0 \qquad \forall (i,j,k,l) \in \mathrm{M}\times\mathrm{N} \\
& x_{ij} \leq 1 \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N} \\
& x_{ij} \geq 0 \qquad \forall (i,j) \in \mathrm{M}\times\mathrm{N}
\end{aligned}
$$

4.5 Résolution du problème par les deux approches

On résout le problème en considérant 2 instances différentes :

— une **première instance** qui utilise la matrice des populations attendues du Tableau 15 et telle que : $m = n = 10,\ w_1 = 1,\ w_2 = 5,\ l = 3,\ g = 1.26157$

— une **deuxième instance** qui utilise la matrice des populations attendues du Tableau 16 et telle que : $m = n = 5,\ w_1 = 2,\ w_2 = 1,\ l = 3,\ g = 1.26157$

84	68	97	98	64	89	82	71	74	76
87	83	98	75	60	90	78	67	92	94
84	68	70	81	67	61	73	92	86	90
79	62	86	79	73	84	76	98	84	90
62	72	66	72	92	80	71	91	87	70
85	77	63	93	90	94	76	81	99	98
76	63	66	84	94	93	72	92	79	65
76	63	92	69	60	88	79	93	66	73
92	82	77	72	77	81	89	95	80	80
88	89	83	86	69	78	91	64	94	92

TABLE 15 – Matrice des t_{ij} utilisée dans la première instance

10	10	10	1	10
10	10	1	1	10
10	10	1	10	10
1	10	10	10	10
1	10	10	10	10

TABLE 16 – Matrice des t_{ij} utilisée dans la deuxième instance

On obtient les résultats visibles sur la Tableau 17 et la Tableau 18 qui **correspondent aux résultats attendus**. À noter que l'on obtient les **mêmes résultats** que ce soit pour :

— avec la **première approche**, qui est un branch and cut réalisée par CPLEX sur **(P1)**

— avec la **deuxième approche**, qui est simplement la résolution de **(P4)**, qui est un programme linéaire en variables continues.

Un **récapitulatif des données voulues** est donné dans le Tableau 19 pour la **première approche** et dans le Tableau 20 pour la **deuxième approche**.

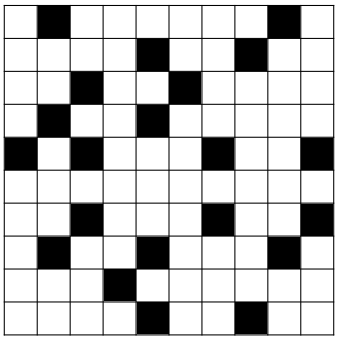

TABLE 17 – Résultats obtenus sur la première instance : **Effectif de l'espèce** e_1 : 6630, **Effectif de l'espèce** e_2 : 317.91564, **Nombre de parcelles non coupées** : 21, **Nombre de côtés appartenant à la lisière** : 84, **Valeur de la fonction économique à l'optimum** : 8219.5782

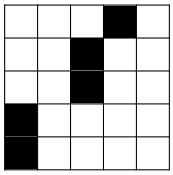

TABLE 18 – Résultats obtenus sur la deuxième instance : **Effectif de l'espèce** e_1 : 191, **Effectif de l'espèce** e_2 : 60.55536, **Nombre de parcelles non coupées** : 5, **Nombre de côtés appartenant à la lisière** : 16, **Valeur de la fonction économique à l'optimum** : 442.55536

	temps (s)	nombre de noeuds
Instance 1	0.01	0
Instance 2	0.01	0

TABLE 19 – Résultats obtenus pour les 2 instances demandées en utilisant **(P1)**

	temps (s)	nombre de noeuds
Instance 1	0.00	0
Instance 2	0.00	0

TABLE 20 – Résultats obtenus pour les 2 instances demandées en utilisant **(P4)**

On remarque que **(P4)** est plus rapide que **(P1)** ce qui était **prévisible** car il est en **variables continues** donc un seul programme linéaire doit être résolu, il n'y a donc **pas de noeuds développé**. Malgré le fait que **(P1)** soit en **variables mixtes** et utilise donc du **branch an cut**, il reste **rapide** car les instances sont de **petites tailles**.

4.6 Résolution du problème par les deux approches avec une contrainte supplémentaire

On résout maintenant la première instance avec une **contrainte supplémentaire** : on impose que le nombre de parcelles non coupées soit supérieur ou égal à 60. Cette contrainte s'écrit de la manière suivante :

$$\sum_{(i,j)\in \mathrm{M}\times\mathrm{N}} x_{ij} >= 60$$

On obtient les résultats visibles sur la Tableau 21 qui **correspondent au résultat attendu**. À noter que l'on obtient les **mêmes résultats** que ce soit pour :

— avec la **première approche**, qui est un branch and cut réalisée par CPLEX sur **(P1)**

— avec la **deuxième approche**, qui est simplement la résolution de **(P4)**, qui est un programme linéaire en variables continues.

Un **récapitulatif des données voulues** est donné dans le Tableau 22 pour la **première approche** et dans le Tableau 23 pour la **deuxième approche**

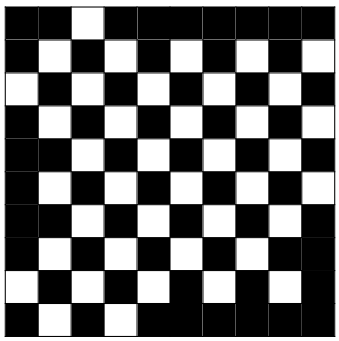

TABLE 21 – Résultats obtenus sur la première instance avec la contrainte supplémentaire instance : **Effectif de l'espèce** e_1 : 3272, **Effectif de l'espèce** e_2 : 696.38664, **Nombre de parcelles non coupées** : 60, **Nombre de côtés appartenant à la lisière** : 184, **Valeur de la fonction économique à l'optimum** : 6753.9332

	temps (s)	nombre de noeuds
Instance 1	0.14	0

TABLE 22 – Résultats obtenus pour l'instance demandée en utilisant **(P1)**

	temps (s)	nombre de noeuds
Instance 1	0.01	0

TABLE 23 – Résultats obtenus pour l'instance demandée en utilisant **(P4)**

La matrice des contraintes perd son caractère **TU** dans le cas général.

4.7 Étude de l'impact de la taille d'une instance sur le temps moyen de calcul

On étude l'**impact de la taille d'une instance sur le temps moyen de calcul des deux approches**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— Les t_{ij} sont choisis aléatoirement de manière uniforme **entre 60 et 100 compris**.

— On garde $w_1 = 1,\ w_2 = 5,\ l = 3,\ g = 1.26157$

— On fait varier les tailles de 10 en 10 en gardant $m = n$ en s'arrêtant à 100.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse donc au temps **moyen** (Figure 12) ainsi qu'à la répartition des valeurs obtenues (Figure 14) en fonction de la taille de l'instance.

On remarque que **plus la taille des instances est importante, plus le temps de calcul en moyenne est élevé et plus les écarts de temps de calcul sont importants**. L'évolution semble **exponentielle**. En passant au log (Figure 13), on observe que l'évolution est presque linéaire. Il aurait fallut faire plus d'expériences pour avoir un résultat plus précis. On peut tout de même imaginer que le problème est **difficile**.

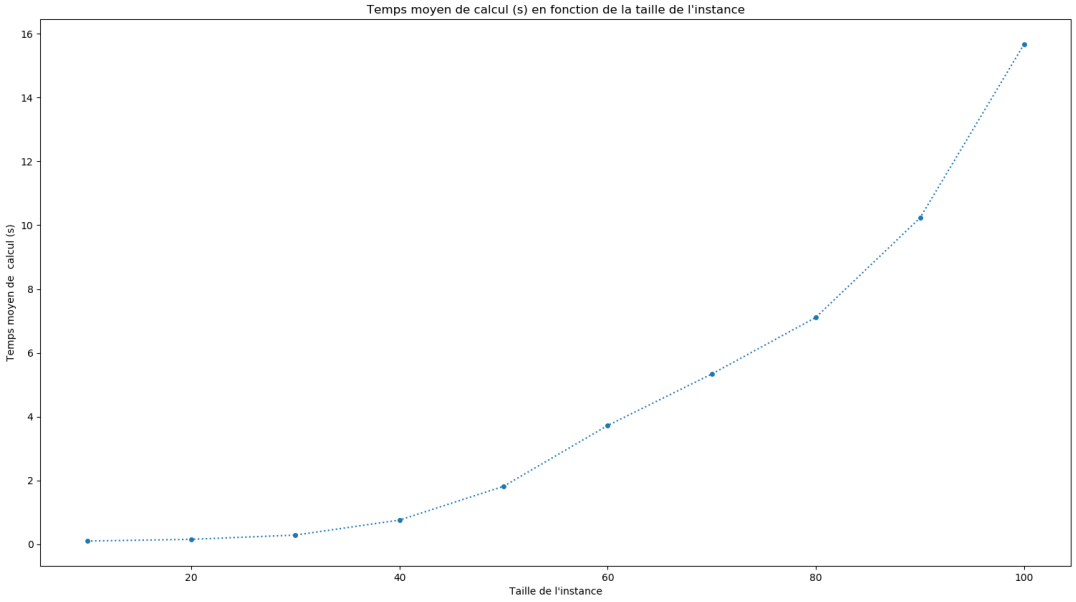

FIGURE 12 – Temps moyen de calcul (s) en fonction de la taille de l'instance

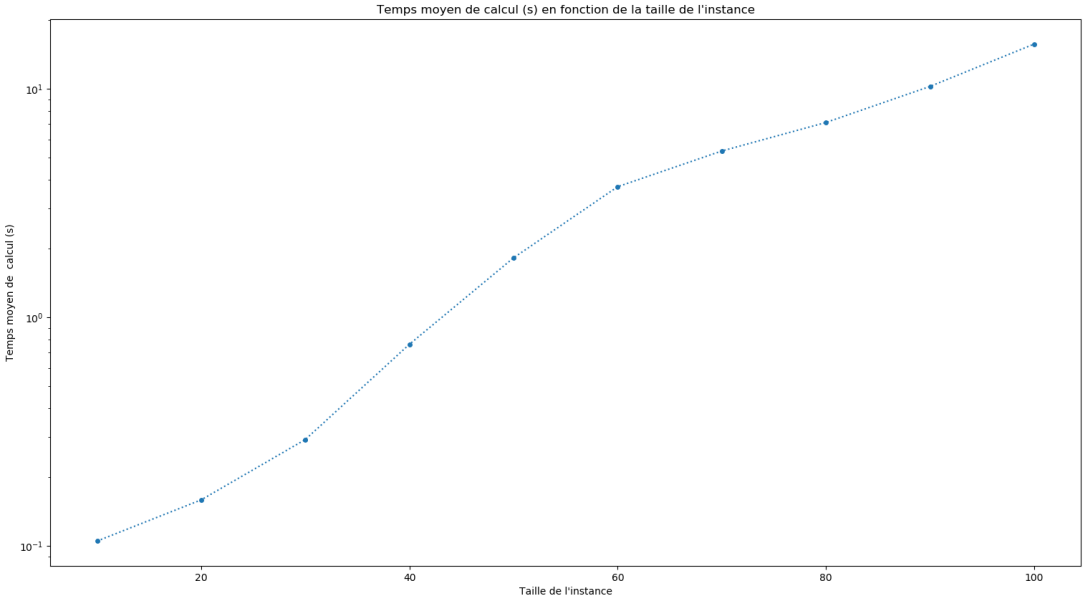

FIGURE 13 – Temps moyen de calcul (s) en fonction de la taille de l'instance

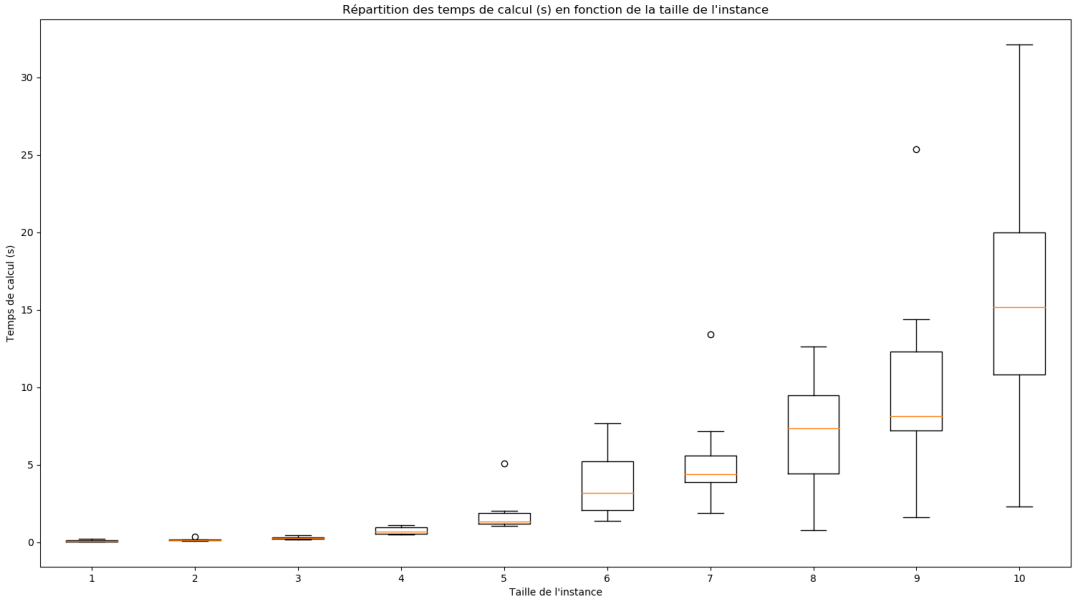

FIGURE 14 – Répartition des temps de calcul (s) en fonction de la taille de l'instance

4.8 Nouveau modèle